Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 70 (2015) 311 – 317

Procedia Computer Science

4th International Conference on Eco-friendly Computing and Communication Systems

A *Trust-based* Uncoordinated Checkpointing Algorithm in Mobile Ad hoc Networks (MANETs)

Poonam Saini[a*] and Shefali Aggarwal[b]

[a,b]*PEC University of Technology, Chandigarh, 160012, INDIA*

Abstract

The paper presents a *trust-value* based uncoordinated check pointing algorithm in Mobile Ad Hoc Networks (MANETs). We aim to improve the overall check pointing overhead incurred in the execution of recovery protocols. Most check pointing algorithms do not consider the mobility rate of nodes while taking a periodic checkpoint, thereby, resulting in poor utilization of resources and increased latency. However, the proposed trust-value based check pointing scheme captures a check point only after a node has endured certain movements. Hence, whenever a node move from one cluster to another, the node moves from a cluster with high trust value to a cluster with low trust value and vice-versa. Therefore, nodes do not rely on any fixed threshold value in order to take a checkpoint. This is due to dependency of trust value of a node on the previous and current trust value of its cluster. Thus, each time a node moves or change to a new cluster, its *cluster_change_count* is compared with a fixed threshold value. The information, hence, is restrained for communication ifthe count is greater than threshold in order to maintain its safety. The proposed algorithm comprises of three phases, namely, *multi-checkpointing phase*, *trust node evaluation phase* and *recovery phase*. The analysis shows better performance of the proposed protocol over existing mobility- based protocol in terms of probability of recovery of failed nodes, residual energy and simulation time.

Peer-review under responsibility of the Organizing Committee of ICECCS 2015

*Keywords:*Distributed Systems, Coordinated Checkpointing, Uncoordinated Chechpointing, Recovery, Trust, MANETs

* Corresponding author. Tel.: +91-172-2753859
E-mail address:nit.sainipoonam@gmail.com

Peer-review under responsibility of the Organizing Committee of ICECCS 2015
doi:10.1016/j.procs.2015.10.022

1. Introduction

A mobile ad hoc network (MANET) is a continually self-configuring, infrastructure-less network of mobile nodes connected without any link. Such networks may operate by themselves or may be connected to the Internet which may result in a dynamic and autonomous topology[1]. Further, distributed transaction processing in MANETs is a major application that requires huge computing capability wherein the inherent failures may degrade the overall performance[2]. The existing literature consists of protocols that increase reliability and minimize the number of failures that includes group communication and rollback recovery. Here, rollback recovery considers a distributed system as a group of processes that communicate through a wireless network. These processes have access to a steady storage to survive different failures by saving the corresponding recovery information. In case of failure, the processes may recover with the help of saved information from these devices. The recovery information contains the state of the processes known as *check points.* However, message passing systems make rollback recovery more complicated due to inter-dependencies as shown in fig. 1. If failure occurs in any process, the dependencies may force a number of processes to rollback leading to a problem called *rollback propagation.* Under some circumstances, rollback propagation may extend back to the initial state of computation leading to the failure of all computations. The resulting condition is called *domino effect.*

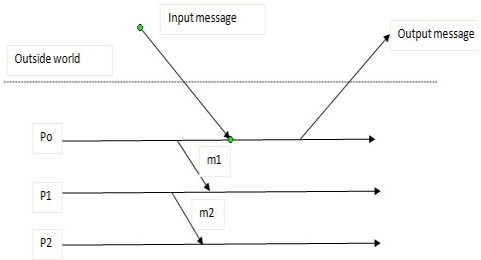

Fig.1. Message passing system

1.1. General Recovery Terms and Definitions

Consistent System State
A *consistent system state* is one in which if a receiver receives a message, then the sender instantly shows the receipt of sent message, *i.e.*, every received message is already sent by sender. The important goal of rollback recovery protocol is to bring a system into consistent state.

Interaction with the Outside World
A message passing system communicates with outside world while taking some input from outside and showing some output to the outside world. It is important that the outside world recognize the system as failure free. Therefore before sending any message the system assures that the state of system is recoverable despite any failure *i.e., output commit proble*m.

In-Transit Message
When a message is being sent and not received at receiver's site, it is called *in-transit* message. If a system assumes communication channels to be reliable, rollback recovery protocols should handle in-transit messages.

Stable Storage
Rollback recovery needs stable storage to save checkpoint data in order to recover from any kind of failures.

Garbage Collection
All checkpointing algorithms consume resources. As the computation increases, the amount of information collected increases. However, most of the data may not be of any use after some time. Therefore, garbage collection should be ensured.

Checkpointing based Rollback Recovery
In distributed algorithms, when failure occurs, rollback recovery restores the state of system to the most recent consistent state. Checkpoint based protocols are easy to implement and imposes fewer restrictions. However, protocols do not ensure that the system is roll backed to pre-failure state. Therefore, checkpoint based rollback recovery is best for systems that communicate with the outside world. Checkpoint based protocols are divided into three sub categories: *Uncoordinated checkpointing, coordinated checkpointing,* and *communication induced checkpointing.*

Uncoordinated Checkpointing
Uncoordinated checkpointing allows a process to take checkpoint anytime without any restriction. However, there is a possibility of *domino effect* which may lost the saved information. Also, the useless checkpoints increase the wastage of space.

Coordinated Checkpointing
In coordinated checkpointing, processes synchronize their checkpoints in order to reach a consistent state. It is less prone to failures and domino effect as each process starts from its latest saved checkpoint after failure. Moreover, coordinated checkpoint ensures that each process saves its checkpoint on stable storage reducing wastage of space resulting in avoiding garbage collection. However this approach has a disadvantage of time delay due to synchronization involved.

Communication Induced Checkpointing
In communication induced checkpointing (CIC), processes takes two type of checkpoints: *local* and *forced*. The local checkpoints can be taken independently, while forced checkpoint *must* be taken to guarantee the eventual progress of the recovery line. In particular, CIC protocols take forced checkpoint to prevent *useless* checkpoints, *i.e.*, checkpoints that do not constitutes a consistent global state[2].

2. Related Work

Arup Acharya and Badrinath, 1994[3] presented a checkpointing protocol in which stable storage of MSS is used by mobile hosts to handle the storage issue. In the approach, each MH takesa checkpoint in three cases,i) before moving into new cluster, ii) before moving away from that cluster and iii) during two-phase rule that describes the checkpoint procedure. Therefore, it ensures that there is no dependency between two checkpoints.

Taesoon Park and H.Y. Yeom, 2000[4] proposed an algorithm in which message logging and dependency tracking is performed by MSS instead of MH in order to handle storage problem. There is no need of coordination amongst mobile hosts and there is no chance of failure. Moreover, the system can handle multiple and parallel failures.

Guohong Cao and Mukesh Singhal, 2001[5] introduced mutable checkpoints. To minimize overheads incurred during coordinated checkpointing such as domino effect, a new scheme called *mutable checkpoints* has been introduced that can be saved anytime and anywhere *i.e.,* local disk or MSS. Thus, mutable checkpoints are beneficial over uncoordinated and coordinated checkpointing due to reduced storage overheads.

Tong- Tony –Chang, 2000[6] presented an efficient recovery algorithm for cluster-based structure based on hybrid structure (checkpointing and rollback recovery). Here, clusters communicate with each other via cluster heads. Each processor maintains and update log to save its state. In case of failure, the process restarts from its last saved state.

Sapna E. George, Ing-Ray Chen, Ying Jin, 2006[7]presented movement based checkpointing and logging for recovery in mobile computing system. In this approach, if a mobile host has changed its cluster a particular number of times called threshold then only checkpoint is taken where, threshold is function of log arrival rate, failure rate, and mobility rate[13]. To calculate this value (threshold), a special model has been designed. Independent checkpointing and message logging is being combined in this protocol enabling asynchronous recovery of a node and also optimize recovery cost, recovery time and storage issues.

A. K Singh-P. K. Jaggi, 2011[8] presented a coordinated checkpointing scheme, called staggered checkpointing, that uses self-stabilizing spanning tree in order to reduce the message overhead, handle dynamic nature of MANETs and reduce resource contention. The protocol supports concurrent checkpoint initiation and successfully handles the overlapping failures in mobile ad hoc networks[13].

Tuli-kumar, 2011[9] proposed a minimum process checkpointing scheme for clustering protocols in MANETs. In the protocol, whenever the cluster heads send routing and other collected information to the base station, it saves the information about the cluster heads. Here, all processes need not to take checkpoint. In case of cluster head failure, a new node is elected as cluster head, thereby, reducing energy consumption and recovery latency.

3. Proposed trust-based checkpointing algorithm

System Model

MANETs consists of *n* number of mobile hosts and *m* number of mobile support stations, where *n* >>*m*. In MANETs, mobile nodes are grouped into clusters, each with a cluster head and gateway nodes. Cluster head ensures communication amongst nodes in same cluster whereas gateway nodes play an intermediate node amongst the nodes belonging to different clusters. There are two type of messages in any cluster namely, *inter*- and *intra*- cluster message. In the scheme, cluster head failure is critical. After its failure, re-election initiates within the cluster. The mobile hosts are connected through wireless links whereas mobile support stations are connected through wired links. The communications links are FIFO. Here, clocks are not synchronized and memory is not shared. In addition, there is an upper bound messages transmission. Mobile support stations are used for stable storage instead of mobile hosts. We use cluster-based approach for better performance. When a node in MANET changes its cluster, only mobile nodes residing in the clusters need to update the information. Hence, storage overhead is greatly reduced.

4. Phases of the proposed protocol

Checkpointing Phase

During checkpointing phase, a mobile host sends and receives computation message to and from other mobile hosts. A mobile host maintains a log of received and sent messages and forwards to current cluster head. The value of the cluster change count threshold when a node moves from one cluster to another cluster is not static, means if the value of cluster change count of a particular node is larger than the value of threshold at that time then the node is said to be prone to attack or unsafe to transfer secured information. A mobile host increments '*cluster_change_value*' on the basis of cluster trust value (as explained later in trust model section) each time the mobile host leaves a cluster and joins. Each mobile host saves a checkpoint independently in the most trusted node (trust factor calculate on the basis of optimal dynamics as explained in trust model) if its '*cluster_change_value*' exceeds a threshold calculated on the basis of trust model. Here, multi checkpointing technique means that whenever a node fails it fetches the last saved checkpoint after reaching the dynamic threshold value. At this time, the node which is at maximum risk of attack can save its data to some trustworthy node of that cluster which is also termed as check-pointing node. Here the '*cluster_change_value'* is always initialized from zero after deleting the previous checkpointing data. Therefore this approach reduced the memory contention and prevents taking of useless checkpoints and also prevents need of garbage collection.

Trust Node Evaluation Phase

Trust of a node is calculated on the basis of trust level of the cluster in which the node is present on a particular instant of time and the cluster change count threshold value. A threshold value is the value which is designed to limit the cluster movements.

$$Threshold = [T \cup (T \cap CH)]/Total\ number\ of\ clusters \quad (1)$$

T-> Number of times a node has been trusted
CH->Number of times a node has been made cluster head

Trust value of cluster member: Trust factor of member nodes of each cluster has been designed on the basis of past interactions between nodes, recommendations from the neighbor nodes and distance between nodes.

$$Recommendation = \mu * t / d \quad (2)$$

μ-> past interaction with this node
t->trust value of recommending node
d->distance between recommending node and recommended node

Moreover trust value of any cluster member is calculated on the basis of this recommendation of all the neighboring nodes and number of trustworthy nodes present in the cluster.

$$Trust\_factor\_value = [\Sigma\ Recommendations/\ n\ -1)] \quad (3)$$

n-> total number of nodes in a cluster

Trust value of the cluster: Now trust of cluster as a whole is being evaluated on the basis of value obtained from the trust value of its member nodes.

$$Trustvalue\_cluster = [\Sigma Trust\_factor\_valueofnodeincluster/(number\ of\ nodes\ in\ a\ cluster)] \quad (4)$$

$$Cluster\_change\_value = previous\ cluster\_change\_value + trustvalue\_cluster * 1 \quad (5)$$

Here, increase in trust value of the nodes increase the trust value of cluster. The value of count that is the value by which the counter must be incremented when a node makes cluster movement is also dynamic and also depends on trust value of both clusters (initial and final), *i.e.*, if the node moves to a cluster having low trust value, then the count value of the cluster is increased by a larger value and vice-versa. Hence, cluster change count and trust value of the cluster are inversely proportional to each other. Also, the cluster head selection is based on the trust value, energy of the node and number of times the node transmits the packets unsuccessfully (which should be minimum).

Recovery Phase

If a node fails or it becomes malicious, the node responsible for recovery sends a signal to each cluster head which further forwards the query to all the other nodes in its cluster. Afterwards, the checkpointed node forwards the required recovery data to the recovery node with optimal and shortest path. The optimal path calculation depends on various factors like the optimal route should contain all trusted nodes that consumes less energy to transfer the data

Pseudo code

1: **Start**
2: RN_i sends request to CH_j for checkpointing data, for all $i \in N$, $j \in$ set of cluster heads
3: Send *Ack*(CH_j, RN_i), where *j* is the checkpointing node, *i* is the recovery node
4: Calculate *Route*(*PR*), where PR is a set of all possible routes calculated by CalculateRoute
5: $t \leftarrow 0$, *initialize time*
6: *InitPopulation P*(*t*), *initialize population with PR*
7: Evaluate *P*(*t*), *evaluate fitness of all nodes individually*
8: **while** (not best) **do**
9: t ← t + 1, *increment time (iteration)*
10: *P*' ← *selectparents P*(*t*), *initialize sub population for offspring production*
11: *Recombine P*'(*t*), *P*'(*t*), *recombination of genes of selected parents with crossover rate 0.5*
12: *Mutate P*'(*t*), *mutate at the rate of* 0.1
13: *Evaluate P*'(*t*), *evaluate new fitness*
14: P ← *CalculateBest P*, *P*'(*t*)
15: **end do**
16: **End**

5. Results and Discussion

Performance Parameters

*Probability of Recovery***:**The probability of a node to be recovered is defined by *probability of recovery*. It depends on the trust value of a node and the cluster change count. If the cluster change count is high and the trust value is low, then the probability of recovery is high.

Residual Energy: Residual energy is the energy left at each node in the network after transmission and reception of packets by the nodes in the network. If the residual energy of a node in the network is less than the threshold, then the node is considered to be faulty or otherwise dead node.

*Trust***:** Trust of a node is used to define the *confidence* on a node participating in the communication over the network. It measures the cluster change count which is the factor responsible for checkpointing of data and recovery.

The following fig. 2 shows the average energy remaining on the nodes in the network plotted against the simulation time. As shown in the figure, the residual energy of the proposed approach is better as compared to the mobility based checkpointing algorithm where the cluster change count is increased by a fixed value. Thus, we can save the energy consumption of nodes in the form of residual energy after applying genetic algorithm on the trusted network.

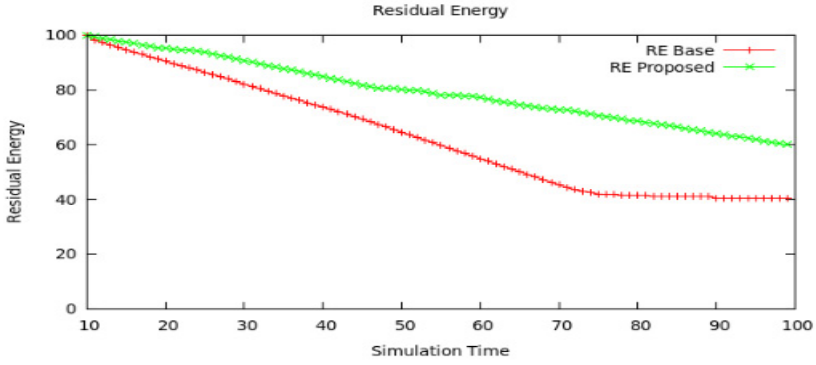

Fig 2.Residual Energy vs. Simulation Time

Fig. 3 shows the graph between residual energy vs probability of recovery. In the proposed protocol, as the residual energy decreases, the probability of recovering the checkpointing data increases when compared to the existing approach[10]. Thus, saving energy consumption of nodes may increase probability of recovery of nodes. The graph shows that route selection by genetic algorithm is optimized for a network having less residual energy.

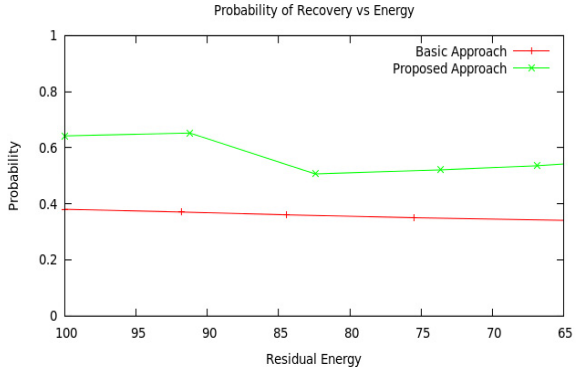

Fig. 3. Probability of recovery vs. Residual energy

Fig. 4 shows the probability of recovery vs trust of a node in the network. The graph shows that as the trust increases, the probability of recovery of nodes in both the approaches also increases. However, the probability of recovering the checkpointing data in the proposed approach is higher in comparison to the existing one. This is due to the selection of optimal route using genetic algorithm through trusted nodes only which is not the case in the existing approach.

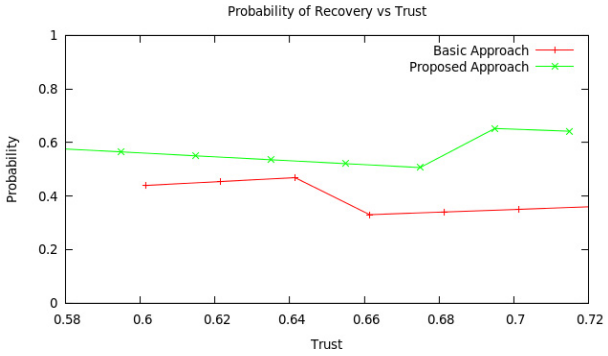

Fig. 4. Probability of Recovery vs. Trust

Conclusion

The paper presented trust-based uncoordinated checkpointing algorithm that prevents useless checkpoints by taking checkpoint only when a node has moved a number of clusters *i.e.*, dynamic in nature. Also, the concept of genetic algorithm has been used for calculating optimal and efficient path between recovering node and the node containing checkpointing data. The analysis shows a significant reduction in useless checkpoints with the use of multi checkpoint scheme. The proposed algorithm also proved to be efficient in terms of probability of recovery and residual energy.

References

1. J.P Macker, S. Corson. Mobile ad hoc networks (MANET): Routing Technology for Dynamic, Wireless Networking. *IEEE Ad Hoc Networking, S. Basagni, M. Conti, S. Giordano, I. Stojmenovic (Eds.)*, Wiley, New York, 2003.
2. Elnozahy, Elmootazbellah Nabil, et al. A survey of rollback-recovery protocols in message-passing systems. *ACM Computing Surveys (CSUR)* 2002, 34: 3, pp. 375-408.
3. Acharya, B.R. Badrinath. Checkpointing Distributed Applications on Mobile Computers. *3rd International Conference on Parallel and Distributed Information Systems* 1994, pp. 73-80.
4. Taesoon Park, Heon Y. Yeom. An asynchronous recovery scheme based on optimistic message logging for mobile computing systems. *20th International Conference on Distributed Computing Systems* 2000, pp. 436-443.
5. Guohong Cao, Mukesh Singhal. Mutable Checkpoints: A New Checkpointing Approach for Mobile Computing Systems. *IEEE Transactions on Parallel and Distributed Systems* 2001, 12: 2, pp 157-172.
6. Tong-Ying Juang et al. An Efficient Asynchronous Recovery Algorithm in Wireless Mobile Ad hoc Networks. *International Conference on Communications in Computing CIC* 2001, pp 143-152.
7. Sapna E. George et al. Movement-Based Checkpointing and Logging for Recovery in Mobile Computing Systems. *Proceedings of MobiDE* 2006, pp. 51-58.
8. A.K Singh, P. K. Jaggi. Staggered Checkpointing and Recovery in Cluster Based Mobile Ad Hoc Networks. *International Conference on Parallel, Distributed Computing technologies and Applications* (PDCTA) 2011, Springer Proceedings, pp. 122-134.
9. R. Tuli, P. Kumar. Minimum process coordinated Checkpointing scheme for ad hoc Networks. *International Journal on AdHoc Networking Systems (IJANS)* 2011, 1: 2, pp. 51-63.
10. Shefali Aggarwal, Poonam Saini. Checkpointing in Mobile Ad Hoc Networks (MANETs)-A survey. *International Journal of Advanced Research in Computer Science and Software Engineering* 2015, 5:3, pp. 488-493.